

---

# Tracking Objects with Higher Order Interactions using Delayed Column Generation

---

**Shaofei Wang**

sfwang0928@gmail.com

**Steffen Wolf**

Heidelberg University, Germany  
steffen.wolf@iwr.uni-heidelberg.de

**Charles C. Fowlkes**

University of California, Irvine  
fowlkes@ics.uci.edu

**Julian Yarkony**

Experian Data Lab, San Diego, CA  
julian.e.yarkony@gmail.com

## Abstract

We study the problem of multi-target tracking and data association in video. We formulate this in terms of selecting a subset of high-quality tracks subject to the constraint that no pair of selected tracks is associated with a common detection (of an object). This objective is equivalent to the classic NP-hard problem of finding a maximum-weight set packing (MWSP) where tracks correspond to sets and is made further difficult since the number of candidate tracks grows exponentially in the number of detections. We present a relaxation of this combinatorial problem that uses a column generation formulation where the pricing problem is solved via dynamic programming to efficiently explore the space of tracks. We employ row generation to tighten the bound in such a way as to preserve efficient inference in the pricing problem. We show the practical utility of this algorithm for tracking problems in natural and biological video datasets.<sup>1</sup>

## 1 Introduction

Multi-target tracking in video is often formulated from the perspective of grouping disjoint sets of candidate detections into “tracks” whose underlying trajectories can be estimated using traditional single-target methods such as Kalman filtering. There is a well developed literature on methods for exploring this combinatorial space of possible data associations in order to find collections of low-cost, disjoint tracks.

We highlight three approaches closely related to our method. Approaches based on reduction to minimum-cost network flow [14] map tracks to unit flows pushed through a network whose edge costs encode track quality. This elegant construction utilizes edge capacity constraints to enforce disjoint tracks and allows for exact, polynomial-time inference. However, this formulation is quite limited in integrating joint statistics over multiple detections assigned to a track. In particular, it is constrained to first-order dynamics in which the cost of a detection being associated with a given track depends only on the immediately neighboring detections.

Multiple Hypothesis Tracking [3, 10, 8] attempts to overcome these limitations by grouping short sequences of detections into a set of hypothesized tracks that can be evaluated and pruned in an online manner. This trades efficiency and global exactness of min-cost flow trackers for additional modeling power. For example, the cost of a track may be computed using, e.g. spline-based fitting of trajectories and instance specific appearance models. However, such methods face a combinatorial

---

<sup>1</sup>This work was supported in part by NSF grant IIS-1253538

problem of assembling compatible sets of tracklets which is usually tackled using greedy approximations.

Our method is most closely related to the Lagrangian relaxation method of [4], which attempts to capture the speed and guarantees min-cost flow tracking while maintaining the modeling advantages of tracklet approaches. A large number of short sequences of detections (subtracks) are generated, each of which is associated with a cost. The set of subtracks form the basis from which tracks are constructed. The corresponding optimization is attacked via sub-gradient optimization of the Lagrangian corresponding to the constrained objective.

Inspired by [11], we attack the problem of reasoning over subtrack assembly constraints using column/row generation [6, 1, 13] to provide faster inference with tighter bounds than [4]. This paper is organized as follows. In Section 2 we formulate tracking as optimization of a linear programming (LP) relaxation equivalent to [4]. In Section 3 we demonstrate a simple case in which the LP relaxation is loose and demonstrate how to tighten the bound. In Section 4 we formulate optimization over the tighter bound and discuss inference using column and row generation. In Section 5 we demonstrate the effectiveness of our approach on pedestrian tracking and biological image data benchmarks.

## 2 Constraint Relaxation for Multi-target Tracking

### 2.1 Feasible Trackings

Given as input a set of candidate detections  $\mathcal{D}$ , each with a specified space-time location, our goal is to identify a collection of tracks that describe the trajectories of objects through a scene and the subset of detections associated with each such track. We assume that a track trajectory is uniquely determined by the set of detections associated with it and that some detections may be false positives not associated with any track.

We denote the set of all possible tracks by  $\mathcal{P}$  and use  $X$  to denote the detection-track incidence matrix  $X \in \{0, 1\}^{|\mathcal{D}| \times |\mathcal{P}|}$  where  $X_{dp} = 1$  if and only if track  $p$  visits detection  $d$ . A solution to the multi-target tracking problem is denoted by the indicator vector  $\gamma \in \{0, 1\}^{|\mathcal{P}|}$  where  $\gamma_p = 1$  indicates that track  $p$  is included in the solution and  $\gamma_p = 0$  otherwise. A collection of tracks specified by  $\gamma$  is a valid solution if and only if each detection is associated with at most one active track. Using  $\Theta \in \mathbb{R}^{|\mathcal{P}|}$  to denote the costs associated with tracks where  $\Theta_p$  describes the cost of each track  $p$ , we express our tracking problem as an integer linear program:

$$\min_{\gamma \in \bar{\Gamma}} \Theta^t \gamma \quad \text{with} \quad \bar{\Gamma} = \{\gamma \in \{0, 1\}^{|\mathcal{P}|} : X\gamma \leq 1\} \quad (1)$$

Here  $\bar{\Gamma}$  is the space of feasible (integer) solutions. We note that this is equivalent to finding a maximum-weight set packing where each set is a collection of detections corresponding to a track and our goal is to choose a collection of pairwise disjoint sets. This problem is NP-hard [7] and this formulation faces further difficulties as the size of the ILP scales exponentially in the number of detections.

### 2.2 Decomposing Track Scores over Subtracks

A classic approach to scoring an individual track is to use a Markov model that incorporates unary scores associated with individual detections along with pairwise comparabilities between subsequent detections assigned to a track. Such an approach provides efficient inference but is limited in its ability to model higher-order dynamical constraints. Instead, we consider a more general scoring function corresponding to a model in which a track is defined by an ordered sequence of subtracks whose scores in turn depend on detections across several frames.

Let  $\mathcal{S}$  denote a set of subtracks, each of which contains  $K$  or fewer detections where  $K$  is a user defined modeling parameter that trades off inference complexity and modeling power. For a given subtrack  $s \in \mathcal{S}$ , let  $s_k$  indicate the  $k$ 'th detection in the sequence  $s = \{s_1, \dots, s_K\}$  ordered by time from earliest (left) to latest (right). We describe the mapping of subtracks to tracks using  $T \in \{0, 1\}^{|\mathcal{S}| \times |\mathcal{P}|}$  where  $T_{sp} = 1$  indicates that track  $p$  contains subtrack  $s$  as a subsequence.

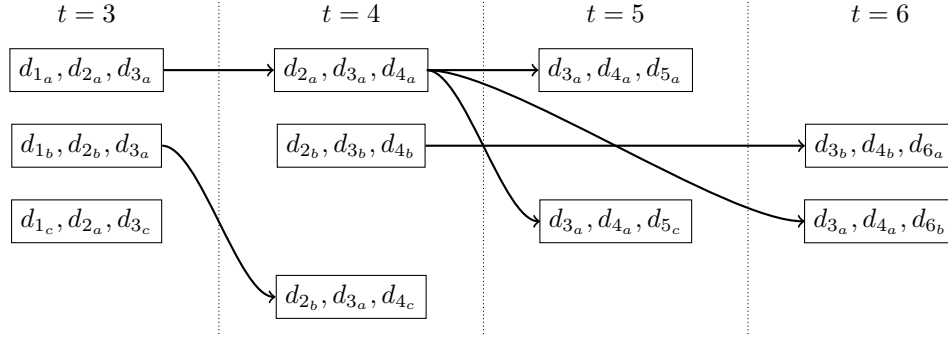


Figure 1: Diagram of detections: We use boxes to denote subtracks and use directed arrows to indicate the valid successors of a given subtrack. Here we associate two indices with a detection. The first index (numbers) describes the time of the detection and the second index (letters) describes the particular observation at that time so  $d_{1a}$  indicates the  $a$ 'th detection at time 1. We order the subtracks by the time of their final detection. Note that a subtrack may skip some time steps (e.g.,  $[d_{3a} d_{4a} d_{6b}]$ ). This corresponds to, e.g., occlusion where there is no detection observed at time 5.

We decompose track costs  $\Theta$  in terms of the subtrack costs  $\theta \in \mathbb{R}^{|S|}$  where each subtrack  $s$  is associated with cost  $\theta_s$  and use  $\theta_0$  to denote a constant cost associated with instancing a track. We define the cost of a track  $p$  denoted  $\Theta_p$  as:

$$\Theta_p = \theta_0 + \sum_{s \in S} T_{sp} \theta_s \quad (2)$$

### 2.3 LP Relaxation and Column Generation

We now attack optimization in Eq 1 using the well studied tools of LP relaxations. We use  $\Gamma = \{\gamma \in [0, 1]^{|P|} : X\gamma \leq 1\}$  to denote a convex relaxation of the constraint set  $\bar{\Gamma}$ . The corresponding relaxed primal and dual problems are written below with dual variables  $\lambda \in \mathbb{R}^{|D|}$ .

$$\min_{\gamma \in \Gamma} \Theta^t \gamma \geq \min_{\gamma \in \Gamma} \Theta^t \gamma = \max_{\substack{\lambda \geq 0 \\ \Theta + X^t \lambda \geq 0}} -1^t \lambda \quad (3)$$

Direct optimization of the dual bound on the right hand side of Eq 3 is still difficult as a consequence of there being one variable in the primal for every possible track in  $\mathcal{P}$  which grows exponentially in the number of detections. Hence we employ a column generation approach [6, 1] that alternates between solving the optimization problem with a small active subset of variables, and identifying inactive variables that may improve the objective then adding these variables to the active subset. Identifying such variables corresponds to finding the most violated constraint (or a set of highly violated constraints including the most violated) in the dual problem and is computed via combinatorial optimization (in our case, dynamic programming).

Alg 1 gives the pseudocode for the column-generation based optimization. Here the nascent subset of primal variables is denoted  $\hat{\mathcal{P}}$ . We use  $\text{COLUMN}(\lambda)$  to indicate a subroutine that identifies a group of violated constraints  $\hat{\mathcal{P}}$  that includes the most violated given  $\lambda$ . Termination occurs when no more violated constraints exist.

### 2.4 Computing $\text{COLUMN}(\lambda)$ using Dynamic Programming

We now discuss how  $\text{COLUMN}(\lambda)$  is computed efficiently for our track cost model using dynamic programming. This is sometimes referred to as the pricing problem in the column generation literature [1].

We specify that a subtrack  $s$  may be preceded by a subtrack  $\hat{s}$  if and only if the least recent  $K - 1$  detections in  $s$  correspond to the most recent  $K - 1$  detections in  $\hat{s}$ . Formally  $s_{k-1} = \hat{s}_k$  for all  $k$  such that  $K \geq k > 1$ . We denote the set of valid subtracks that may precede a subtrack  $s$  as  $\{\Rightarrow s\}$ . This structure is illustrated graphically in Fig 1.

---

**Algorithm 1** Dual Optimization

---

```

 $\hat{P} \leftarrow \{\}$ 
repeat
   $\lambda \leftarrow \arg \max_{\substack{\lambda \geq 0 \\ \Theta_{\hat{P}} + X_{(:, \hat{P})}^T \lambda \geq 0}} -1^t \lambda$ 
   $\dot{P} \leftarrow \text{COLUMN}(\lambda)$ 
   $\hat{P} \leftarrow [\hat{P}, \dot{P}]$ 
until  $|\dot{P}| = 0$ 

```

---



---

**Algorithm 2** Upper Bound Rounding

---

```

while  $\exists p \in \mathcal{P}$  s.t.  $\gamma_p \notin \{0, 1\}$  do
   $p^* \leftarrow \arg \min_{\substack{p \in \mathcal{P} \\ \gamma_p > 0}} \Theta_p \gamma_p - \sum_{\hat{p} \in \mathcal{P}_{\perp p}} \gamma_{\hat{p}} \Theta_{\hat{p}}$ 
   $\gamma_{\hat{p}} \leftarrow 0 \quad \forall \hat{p} \in \mathcal{P}_{\perp p^*}$ 
   $\gamma_{p^*} \leftarrow 1$ 
end while
RETURN  $\gamma$ 

```

---

Figure 2: (Left): Algorithm for dual-optimization of a lower bound on the optimal tracking by column generation where the notation  $X_{(:, \hat{P})}$  denotes selection of a subset of columns of  $X$ . (Right) We compute upper-bounds on the optimal tracking using a rounding procedure which greedily selects primal variables  $\gamma$  while removing intersecting tracks. This same upper bound procedure is also used during column/row generation in Alg 3.

To permit the use of tracks with less than  $K$  detections, subtracks with fewer than  $K$  detections are expanded to size  $K$  by padded them with “no-observation”; denoted “0”, on the left side of the subtrack. For example consider a subtrack  $s = \{s_1, s_2\}$  where  $K = 5$ . After padding  $s$  with “no-observation” detections we write  $s$  as  $s = \{0, 0, 0, s_4, s_5\}$  where  $s_1$  corresponds to  $s_4$  and  $s_2$  corresponds to  $s_5$ . For each “no-observation” we create a corresponding zero valued  $\lambda$  term.

We use  $\ell \in \mathbb{R}^{|\mathcal{S}|}$  to denote the *cost to go* computed during dynamic programming. Here  $\ell_s$  is the cost of the cheapest track that terminates at subtrack  $s$ . Ordering subtracks by the time of last detection allows efficient computation of  $\ell$  using the following dynamic programming update:

$$\ell_s \leftarrow \theta_s + \lambda_{s_K} + \min \left\{ \min_{\hat{s} \in \{\Rightarrow s\}} \ell_{\hat{s}}, \quad \theta_0 + \sum_{k=0}^{K-1} \lambda_{s_k} \right\} \quad (4)$$

We find it is useful to add not only the minimum cost track (most violated constraint) to  $\hat{P}$  but also the (most violating) track terminating at each possible subtrack. This set of tracks is easy to extract from the dynamic program since it stores the minimum cost track terminating at each subtrack. Only tracks corresponding to violated constraints are added to  $\hat{P}$ . While this over-generation of constraints substantially increases the number of constraints in the dual, we find that many of these constraints prove to be useful in the final optimization problem (similar behavior has been observed in [13]). Additionally, in our implementation dynamic programming consumes the overwhelming majority of computation time so adding more columns per iteration yielded faster overall run time.

## 2.5 Rounding Fractional Solutions

We compute upper bounds using a fast principled method that avoids resolving the LP [4]. Observe that each solution of the LP during the column generation process (Alg 1) corresponds to a (fractional) primal solution in addition to the dual solution (computed “for free” by many LP solvers when solving the dual). We attack rounding a fractional  $\gamma$  via a greedy iterative approach that, at each iteration, selects the track  $p$  with minimum value  $\Theta_p \gamma_p$  discounted by the fractional cost of any tracks that share a detection with  $p$  (and hence can no longer be added to the tracking if  $p$  is added). We write the rounding procedure in Alg 2 using the notation  $\mathcal{P}_{\perp p}$  to indicate the set of tracks in  $\mathcal{P}$  that intersect track  $p$  (excluding  $p$  itself).

## 2.6 Anytime Lower Bounds

It is useful in practice to be able to compute a lower bound on the original objective during the optimization procedure (i.e., prior to adding all the violated columns to the dual). In Appendix A.1

we show that it is possible to compute such an anytime lower bound using the following formula.

$$\min_{\gamma \in \Gamma} \Theta^t \gamma \geq -1^t \lambda + \sum_{d \in \mathcal{D}} \min\{0, \min_{\substack{s \in \mathcal{S} \\ s_K = d}} \ell_s\} \quad \forall \lambda \geq 0 \quad (5)$$

The lower bound computed in Eq 5 is maximized at termination of Alg 1 with value equal to the relaxation in Eq 3. This is because no violated constraints in the dual exist at termination and hence  $\ell_s \geq 0$  for all  $s \in \mathcal{S}$ . Empirically the bound increases as a function of optimization time.

### 3 Tightening the Bound

Our original LP relaxation only contains constraints for collections of tracks that share a common detection. From the view point of maximum-weight set packing, this includes some cliques of conflicting sets but misses many others.

#### 3.1 Fractional solutions from mutually exclusive triplets

As a concrete example, we consider a case where the LP relaxation Eq 3 provides a loose lower bound which is visualized in Fig 3. Consider four tracks  $\mathcal{P} = \{p_1, p_2, p_3, p_4\}$  over three detections  $\mathcal{D} = \{d_1, d_2, d_3\}$  where the first three tracks each contain two of three detections  $\{d_1, d_2\}, \{d_1, d_3\}, \{d_2, d_3\}$ , and the fourth track contains all three  $\{d_1, d_2, d_3\}$ . Suppose the track costs are given by  $\Theta_{p_1} = \Theta_{p_2} = \Theta_{p_3} = -4$  and  $\Theta_{p_4} = -5$ . The optimal integer solution sets  $\gamma_{p_4} = 1$ , and has a cost of  $-5$ . However the optimal fractional solution sets  $\gamma_{p_1} = \gamma_{p_2} = \gamma_{p_3} = 0.5$ ;  $\gamma_{p_4} = 0$  which has cost  $-6$ . Hence the LP relaxation is loose in this case. Even worse, rounding the fractional solution results in a sub-optimal solution.

#### 3.2 Tightening the Bound over Triplets of Detections

A tighter bound can be motivated by the following observation. For any set of three unique detections the number of tracks that pass through two or more members can be no larger than one. Thus the following inequality holds for groups of three unique detections (which we refer to as triplets)  $d_1, d_2, d_3$ . We use [...] to express the indicator function.

$$\sum_{p \in \mathcal{P}} [X_{pd_1} + X_{pd_2} + X_{pd_3} \geq 2] \gamma_p \leq 1 \quad (6)$$

We now apply our tighter bound to tracking. We denote the set of triplets as  $\mathcal{C}$  and index it with  $c$ . We denote the subset of  $\Gamma$  that satisfy the inequalities in Eq 6 as  $\Gamma^C$  and define it using a constraint matrix  $C \in \{0, 1\}^{|\mathcal{C}| \times |\mathcal{P}|}$ .

$$\Gamma^C : \{\gamma \in \mathbb{R}^{|\mathcal{P}|} : \gamma \geq 0, \quad X\gamma \leq 1, \quad C\gamma \leq 1\} \quad (7)$$

The constraint matrix has a row for each conflicting triplet specified as follows.

$$C_{cp} = [\sum_{d \in c} X_{dp} \geq 2] \quad \forall c \in \mathcal{C}, p \in \mathcal{P}$$

### 4 Optimization over $\Gamma^C$

We write tracking as optimization in the primal and dual form below.

$$\min_{\gamma \in \Gamma^C} \Theta^t \gamma = \max_{\substack{\lambda \geq 0 \\ \lambda^C \geq 0 \\ \Theta + X^t \lambda + C^t \lambda^C \geq 0}} -1^t \lambda - 1^t \lambda^C \quad (8)$$

Given that  $\mathcal{P}$  and  $\mathcal{C}$  are of enormous size we use column and row generation jointly. The nascent subsets of  $\mathcal{P}, \mathcal{C}$  are denoted  $\hat{\mathcal{P}}, \hat{\mathcal{C}}$  respectively. We write column/row generation optimization given subroutines COLUMN( $\lambda, \lambda^C$ ), ROW( $\gamma$ ) that identify a group of violated constraints in primal and

dual including the most violated in each. Generating rows is done via exhaustive search and discussed in Section 4.1. Generating columns is performed using a fast branch and bound procedure where bounding is done using dynamic programming and is discussed in Section 4.2. We denote the violated columns and rows identified as  $\hat{\mathcal{P}}, \hat{\mathcal{C}}$  respectively. We write the column/row generation optimization in Alg 3. Any time upper/lower bounds are produced using the methods in Sections 2.5 and 2.6 respectively. Lower-bound computation is modified from Eq 5 in Section A.2.

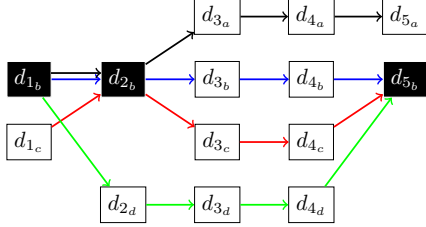


Figure 3: We depict a case where four tracks conflict over a triplet of detections. The relevant triplet is  $d_{1b}, d_{2b}, d_{5b}$  with each colored flow corresponds to a track described in Section 3. Specifically, blue flow corresponds to  $p_4$  while other colored flows correspond to  $p_1, p_2$  and  $p_3$  respectively. Triplets may refer detections highly separated in time though this is not depicted in the picture above.

---

**Algorithm 3** Column/Row Generation

---

```

 $\hat{\mathcal{P}} \leftarrow \{\}, \quad \hat{\mathcal{C}} \leftarrow \{\}$ 
repeat
     $\max_{\lambda \geq 0, \lambda^c \geq 0} -1^t \lambda - 1^t \lambda^c$ 
     $\Theta_{\hat{\mathcal{P}}} + X_{(\cdot, \hat{\mathcal{P}})}^t \lambda + C_{(\hat{\mathcal{C}}, \hat{\mathcal{P}})}^t \lambda^c \geq 0$ 
    Recover  $\gamma$  from  $\lambda$  (provided by LP solver)
     $\hat{\mathcal{P}} \leftarrow \text{COLUMN}(\lambda, \lambda^c)$ 
     $\hat{\mathcal{C}} \leftarrow \text{ROW}(\gamma)$ 
     $\hat{\mathcal{P}} \leftarrow [\hat{\mathcal{P}}, \hat{\mathcal{P}}]$ 
     $\hat{\mathcal{C}} \leftarrow [\hat{\mathcal{C}}, \hat{\mathcal{C}}]$ 
until  $\hat{\mathcal{P}} = \{\}$  and  $\hat{\mathcal{C}} = \{\}$ 

```

---

#### 4.1 Row Generation

Finding the most violated row consists of the following optimization.

$$\max_{c \in \mathcal{C}} \sum_{p \in \mathcal{P}} C_{cp} \gamma_p \quad (9)$$

Enumerating  $\mathcal{C}$  is unnecessary and we generate its rows as needed by considering only triplets  $c = \{d_{c_1} d_{c_2} d_{c_3}\}$  such that for each of pair  $d_{c_i}, d_{c_j}$  there exists a fractional valued track containing both  $d_{c_i}$  and  $d_{c_j}$ .

We find experimentally that adding only the most violated row is efficient, and we only add a row when no violated columns exist. However in other domains/data sets it may be beneficial to add many violated rows at once and add them even when violated columns exist.

#### 4.2 Generating Columns under Triplet Constraints

We denote the value of the slack corresponding to an arbitrary column  $p$  as  $V(\Theta, \lambda, \lambda^c, p)$  and the most violated as  $V^*(\Theta, \lambda, \lambda^c)$  which we define below.

$$V(\Theta, \lambda, \lambda^c, p) = \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp} \quad (10)$$

$$V^*(\Theta, \lambda, \lambda^c) = \min_{p \in \mathcal{P}} V(\Theta, \lambda, \lambda^c, p)$$

Solving for  $V^*(\Theta, \lambda, \lambda^c)$  can not be directly attacked using dynamic programming as in Section 2.4. However dynamic programming can be applied if we ignore the triplet term  $\sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp}$ , providing a lower bound.

This invites a branch and bound (B&B) approach. We find B&B is very practical because experimentally we observe that the number of non-zero values in  $\lambda^c$  at any given iteration is small ( $< 5$ ) for real problems. The set of branches in our B&B tree is denoted  $\mathcal{B}$ . Each branch  $b \in \mathcal{B}$  is defined by two sets  $\mathcal{D}_{b+}$  and  $\mathcal{D}_{b-}$ . These correspond to detections that must be included on the track and those that must not be included on the track respectively. We write the set of all tracks that are consistent with a given  $\mathcal{D}_{b-}$ ,  $\mathcal{D}_{b+}$  or consistent with both  $\mathcal{D}_{b-}$  and  $\mathcal{D}_{b+}$  as  $\mathcal{P}_{b-}$ ,  $\mathcal{P}_{b+}$  and  $\mathcal{P}_{b\pm}$  respectively. We specify the bounding, branching, and termination operators in Sections 4.2.1, 4.2.2, 4.2.3 respectively. The initial branch  $b$  is defined by  $\mathcal{D}_{b+} = \mathcal{D}_{b-} = \{\}$ .

#### 4.2.1 Bounding Operation

Let  $V^b(\Theta, \lambda, \lambda^c)$  denote the value of the most violating slack over columns in  $\mathcal{P}_{b\pm}$ . We can compute a lower-bound for this value, denoted  $V_{lb}^b$  by independently optimizing the dynamic program and the triplet penalty.

$$\begin{aligned}
V^b(\Theta, \lambda, \lambda^c) &= \min_{p \in \mathcal{P}_{b\pm}} V(\Theta, \lambda, \lambda^c, p) \\
&= \min_{p \in \mathcal{P}_{b\pm}} \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp} \\
&\geq \min_{p \in \mathcal{P}_{b-}} \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \min_{p \in \mathcal{P}_{b+}} \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp} \\
&\geq \min_{p \in \mathcal{P}_{b-}} \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c [\sum_{d \in c} [d \in \mathcal{D}_{b+}] \geq 2] \\
&= V_{lb}^b(\Theta, \lambda, \lambda^c)
\end{aligned} \tag{11}$$

Observe that dynamic programming can be used to efficiently search over  $\mathcal{P}_{b-}$  to minimize the first term. For efficiency, subtracks whose inclusion conflicts with any detection in the required set  $\mathcal{D}_{b+}$  can easily be removed before running the dynamic program.

#### 4.2.2 Branch Operation

We now consider the branch operation. We describe an upper bound on  $V^b(\Theta, \lambda, \lambda^c)$  as  $V_{ub}^b(\Theta, \lambda, \lambda^c)$ . This is constructed by adding in the active  $\lambda^c$  terms ignored when constructing  $V_{lb}^b(\Theta, \lambda, \lambda^c)$ . Let  $p_b = \arg \min_{p \in \mathcal{P}_{b-}} \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp}$ .

$$\begin{aligned}
V_{ub}^b(\Theta, \lambda, \lambda^c) &= V_{lb}^b(\Theta, \lambda, \lambda^c) + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp_b} [\sum_{d \in c} [d \in \mathcal{D}_{b+}] < 2] \\
&= \Theta_{p_b} + \sum_{d \in \mathcal{D}} \lambda_d X_{dp_b} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c [\sum_{d \in c} [d \in \mathcal{D}_{b+}] \geq 2] + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp_b} [\sum_{d \in c} [d \in \mathcal{D}_{b+}] < 2] \\
&= V(\Theta, \lambda, \lambda^c, p_b) + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c [\sum_{d \in c} [d \in \mathcal{D}_{b+}] \geq 2] - \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp_b} [\sum_{d \in c} [d \in \mathcal{D}_{b+}] \geq 2] \\
&= V(\Theta, \lambda, \lambda^c, p_b) + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^c (1 - C_{cp_b}) [\sum_{d \in c} [d \in \mathcal{D}_{b+}] \geq 2] \\
&\geq V(\Theta, \lambda, \lambda^c, p_b) \\
&\geq V^b(\Theta, \lambda, \lambda^c)
\end{aligned} \tag{12}$$

Now consider the largest triplet constraint term  $\lambda_c^c$  that is included in  $V_{ub}^b(\Theta, \lambda, \lambda^c, p_b)$  but not  $V_{lb}^b(\Theta, \lambda, \lambda^c)$ .

$$c^* \leftarrow \arg \max_{c \in \hat{\mathcal{C}}} \lambda_c^c C_{cp_b} [\sum_{d \in c} [d \in \mathcal{D}_{b+}] < 2] \tag{13}$$

We create eight new branches for each of the eight different ways of splitting the detections in the triplet corresponding to  $c^*$  between the include (+) and exclude (−) sets. In Fig 4 we enumerate the splits for a triplet of detections  $c = \{d_1, d_2, d_3\}$ . In Section 4.2.3 we establish that if  $b$  is the lowest cost branch and  $\lambda_{c^*}^c = 0$  then  $p_b$  is the track corresponding to the most violated column in  $\mathcal{P}$ . Hence a branch operator is not applied if  $\lambda_{c^*}^c = 0$ . We refer to a branch  $b$  such that  $\lambda_{c^*}^c = 0$  as terminating.

#### 4.2.3 Establishing Optimality at Termination

We now establish that B&B produces the most violated column at termination. We do this by proving that the cost of the track corresponding to the lowest cost branch is both an upper and lower bound on  $V^*(\Theta, \lambda, \lambda^c)$  if that branch is terminating.

Child	-	+	Child	-	+	Child	-	+	Child	-	+
$D_{b_1}$	$\{\}$	$d_1 d_2 d_3$	$D_{b_2}$	$d_1$	$d_2 d_3$	$D_{b_3}$	$d_2$	$d_1 d_3$	$D_{b_4}$	$d_1 d_2$	$d_3$
$D_{b_5}$	$d_3$	$d_1, d_2$	$D_{b_6}$	$d_1, d_3$	$d_2$	$D_{b_7}$	$d_2 d_3$	$d_1$	$D_{b_8}$	$d_1 d_2 d_3$	$\{\}$

Figure 4: We enumerate the eight sets each describing one way of partitioning the three detections  $d_1 d_2, d_3$  between the include (+) and exclude (-) sets for the children of branch  $b$ . For example, branch  $D_{b_4}$  excludes  $d_1$  and  $d_2$  but includes  $d_3$  so  $D_{b_4-} = [D_{b-} \cup d_1 \cup d_2]$  and the set  $D_{b_4+} = [D_{b+} \cup d_3]$ .

Consider branch  $b^*$  with corresponding track  $p_{b^*}$  such that the following conditions are true: (1)  $b^*$  is the lowest cost branch in the B&B tree; (2)  $b^*$  is terminating. We write criteria (1),(2) formally below.

$$(1) \quad V_{lb}^{b^*}(\Theta, \lambda, \lambda^c) \leq V_{lb}^b(\Theta, \lambda, \lambda^c) \quad \forall b \in \mathcal{B} \quad (14)$$

$$(2) \quad 0 = \max_{c \in \tilde{\mathcal{C}}} \lambda_c^c C_{cp_{b^*}} \left[ \sum_{d \in c} [d \in \mathcal{D}_{b^*+}] < 2 \right] \quad (15)$$

We now establish that  $V(\Theta, \lambda, \lambda^c, p_{b^*}) = V^*(\Theta, \lambda, \lambda^c)$ . Recall that by definition of B&B that the lowest value bound in any B&B tree is a lower bound on the true solution. Therefor Eq 14 implies the following.

$$V^*(\Theta, \lambda, \lambda^c) \geq V_{lb}^{b^*}(\Theta, \lambda, \lambda^c) \quad (16)$$

We now plug Eq 15 into Eq 12, and deduce the following:

$$\begin{aligned} V_{lb}^{b^*}(\Theta, \lambda, \lambda^c) &\geq V(\Theta, \lambda, \lambda^c, p_{b^*}) \\ &\geq V^*(\Theta, \lambda, \lambda^c) \end{aligned} \quad (17)$$

Observe that Eq 16 establishes that  $V^*(\Theta, \lambda, \lambda^c) \geq V_{lb}^{b^*}(\Theta, \lambda, \lambda^c)$ . Therefore all inequalities in Eq 17 are equalities and hence  $p_{b^*}$  is the lowest cost track. As in Section 2.4 dynamic programming facilitates the addition of many columns per iteration of Alg 3. Any track  $p$  produced during calls to dynamic programming during B&B such that  $V(\Theta, \lambda, \lambda^c, p) < 0$  should be added to the set  $\tilde{\mathcal{P}}$  in Alg 3.

## 5 Experiments

### 5.1 Experiments on the Particle Tracking Challenge Data

We applied our algorithm to the data from the Particle Tracking Challenge (PTC) simulated data set for high density microtubules (SNR7), which consists of two videos (train, test) of 99 images of  $512 \times 512$  pixels over time where the test data set contains 6733 tracks that cover 71035 detections. We take the set of ground truth detections as the set of detections and apply our tracking algorithm.

From the set of detections we generate a matrix of possible subtracks of  $K$  detections as follows. Consider a directed graph where the nodes are the set of all detections. For each detection  $d$  we draw a line from  $d$  to each of its three spatial nearest neighbors in the following frame. The set of all paths containing  $K$  nodes in this graph is the set of subtracks. In total we find 1,873,341 subtracks. We are provided with costs for each subtrack via logistic regression based on motion features consisting of autocorrelation, and autocovariance, and other distance features.

With  $K = 3$  and optimized hyper-parameters we reach a Jaccard score of 0.924 as compared to a baseline of 0.754. We identified 6329 tracks that are in the ground truth, missed 404 tracks that are in the ground truth and identified 118 tracks that are not in the ground truth. These results are produced via providing our output to the benchmarking code associated with [5]. In Fig 5, we applied Alg 1 and study tracking performance in terms of accuracy and cost.

### 5.2 Tracking Pedestrians in Video

We use a part of MOT 2015 training set [9] to train and evaluate real-world tracking models. MOT dataset consists of popular pedestrian benchmark datasets such TUD, ETH and PETS. Specifically



we use the learning framework of [12] with Kalman Filters to train models using ETH-Sunnyday and TUD-Stadtmitte, and test the models on TUD-Campus sequence. For detections we use the raw detector output provided by the MOT dataset. We train the models with varying tracklet length ( $K = 2, 3, 4$ ) and allow for occlusion up to three frames. There are altogether 71 frames and 322 detections in the video, numbers of subtracks are 1,068, 3,633 and 13,090 for  $K = 2, 3, 4$ . For  $K = 2$  we observe 48.5% Multiple Object Tracking Accuracy [2], 11 identity switches and 9 track fragments or for short hand (48.5,11,9). However when setting  $K = 3, 4$  the performance is (49,10,7), and (49.9,9,7) which constitutes noticeable improvements over all three metrics. In Fig 5 we compared the timing/cost performance of our algorithm with the baseline algorithm of [4] on problem instances with a loose lower bound.

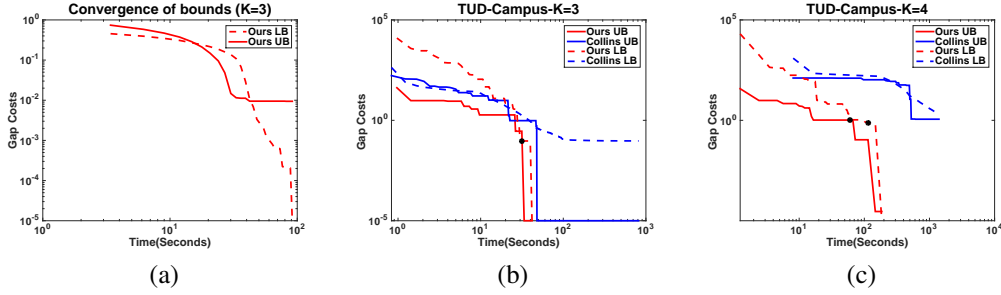


Figure 5: (a): For  $K = 3$  on Particle Tracking Challenge data, we show the convergence of the bounds as a function of time in seconds on the test data set. We display the value of the upper, and lower bounds. (b) and (c): when training on a subset of motion features on MOT dataset we get instances with loose bound. For the two examples we plot the gap (absolute value of the difference) between the bounds and the final lower bound as a function of time. We indicate each time that a triplet is added with a black dot on the lower bound plot. In all examples the bound of [4] is loose and at least one triplet is needed to produce a tight bound which results in visually compelling trackings. We find that additional features for the structured SVM results in a tighter bound in practice. In both comparisons against [4] our upper- and lower- bounds are tight at termination.

## 6 Conclusions

We have introduced a new method for multi-target tracking built on an LP relaxation of the maximum-weight set packing problem. Our core contribution is a column generation approach that exploits dynamic programming to generate a large number of candidate tracks concurrently. This yields an efficient algorithm and provides rigorous bounds that can be tightened via row generation. We empirically observe that our algorithm rapidly produces compelling tracking results along with strong anytime performance relative the baseline Lagrangian relaxation[4].



Figure 6: We illustrate a qualitative example of improvement as a result of increasing subtrack length. Top row is detector output and associated confidence provided by [9]. Second row and third row correspond to trackers of subtrack length  $K = 2$  and  $K = 4$  respectively. Notice that for  $K = 2$  track 1 changes identity to 5, while with  $K = 4$  the identity of track 1 does not change. Missing detections in tracking results are interpolated linearly and tracks are smoothed after interpolation.

## A Lower Bounds on the Optimal Tracking Cost

### A.1 For Relaxation over $\Gamma$

We now study lower bounds on the integer programming objective for tracking which we re-write below using Lagrange multipliers  $\lambda$  to enforce the constraints defining  $\bar{\Gamma}$ .

$$\min_{\gamma \in \Gamma} \Theta^t \gamma = \min_{\gamma \in \{0,1\}^{|\mathcal{P}|}} \max_{\lambda \geq 0} \Theta^t \gamma + \lambda^t (X\gamma - 1) \quad (18)$$

We now add the redundant constraint that no two tracks terminate at the same detection. We use  $\mathcal{P}^d$  to refer to the set of tracks terminating at detection  $d$ .

$$\text{Eq 18} = \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1 \quad \forall d \in \mathcal{D}}} \max_{\lambda \geq 0} \Theta^t \gamma + \lambda^t (X\gamma - 1) \quad (19)$$

We now relax the optimization and consider any non-negative  $\lambda$ . Recall that every time dual optimization (during Alg 1,3) is solved a non-negative  $\lambda$  term is produced.

$$\begin{aligned} \text{Eq 18} &\geq \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1 \quad \forall d \in \mathcal{D}}} \Theta^t \gamma + \lambda^t (X\gamma - 1) \\ &= -\lambda^t 1 + \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1 \quad \forall d \in \mathcal{D}}} (\Theta^t + \lambda^t X) \gamma \\ &= -\lambda^t 1 + \sum_{d \in \mathcal{D}} \min\{0, \min_{p \in \mathcal{P}^d} \Theta_p + \sum_{d \in \mathcal{D}} X_{dp} \lambda_d\} \\ &= -\lambda^t 1 + \sum_{d \in \mathcal{D}} \min\{0, \min_{\substack{s \in \mathcal{S} \\ s_K = d}} \ell_s\} \end{aligned} \quad (20)$$

At termination of column generation no violated constraints exist so  $\ell_s \geq 0 \forall s \in \mathcal{S}$  and thus the lower bound has value identical to the LP relaxation over  $\Gamma^C$  in Eq 3.

## A.2 For Relaxation over $\Gamma^C$

We now consider computing an anytime lower bound on the optimal tracking when  $\lambda^C$  terms are present using a similar procedure to that in Section A.1.

$$\begin{aligned} \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \gamma \in \Gamma^C \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1}} \Theta^t \gamma &= \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1}} \max_{\substack{\lambda^c \geq 0 \\ \lambda \geq 0}} \Theta^t \gamma + \lambda^t (X\gamma - 1) + \lambda^{Ct} (C\gamma - 1) \\ &= \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1}} \max_{\lambda^c \geq 0} -\lambda^t 1 - \lambda^{Ct} 1 + \Theta^t \gamma + \lambda^t X\gamma + \lambda^{Ct} C\gamma \end{aligned} \quad (21)$$

We now relax the optimization and consider any non-negative  $\lambda, \lambda^C$ .

$$\begin{aligned} \text{Eq 21} &\geq -\lambda^t 1 - \lambda^{Ct} 1 + \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1}} \Theta^t \gamma + \lambda^t X\gamma + \lambda^{Ct} C\gamma \\ &\geq -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, \min_{p \in \mathcal{P}^d} \Theta_p\} + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^C C_{cp} \end{aligned} \quad (22)$$

For short hand we define the following quantity  $V^{d*}(\Theta, \lambda, \lambda^C)$  as the lowest cost over tracks terminating at detection  $d$ .

$$V^{d*}(\Theta, \lambda, \lambda^C) = \min_{p \in \mathcal{P}^d} \Theta_p + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^C C_{cp} \quad (23)$$

We rewrite the final expression in Eq 22 as follows.

$$\begin{aligned} -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, \min_{p \in \mathcal{P}^d} \Theta_p\} + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^C C_{cp} \\ = -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, V^{d*}(\Theta, \lambda, \lambda^C)\} \end{aligned} \quad (24)$$

We bound  $V^{d*}(\Theta, \lambda, \lambda^C)$  from below in two different ways. First, we ignore the  $\lambda^C$  terms and optimize via dynamic programming producing the following bound.  $V^{d*}(\Theta, \lambda, \lambda^C) \geq \min_{\substack{s \in \mathcal{S} \\ s_K=d}} \ell_s$ .

However we also bound  $V^{d*}(\Theta, \lambda, \lambda^C)$  by the minimizer over all  $d$  hence  $V^{d*}(\Theta, \lambda, \lambda^C) \geq V^*(\Theta, \lambda, \lambda^C)$ . Combining the two bounds on  $V^{d*}(\Theta, \lambda, \lambda^C)$  we produce the following bound.

$$V^{d*}(\Theta, \lambda, \lambda^C) \geq \max\{V^*(\Theta, \lambda, \lambda^C), \min_{\substack{s \in \mathcal{S} \\ s_K=d}} \ell_s\} \quad (25)$$

We now apply the bound in Eq 25 to produce an anytime computable lower bound on the optimal tracking.

$$\begin{aligned} \min_{\substack{\gamma \in \{0,1\}^{|\mathcal{P}|} \\ \gamma \in \Gamma^C \\ \sum_{p \in \mathcal{P}^d} \gamma_p \leq 1}} \Theta^t \gamma &\geq -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, \min_{p \in \mathcal{P}^d} \Theta_p\} + \sum_{d \in \mathcal{D}} \lambda_d X_{dp} + \sum_{c \in \hat{\mathcal{C}}} \lambda_c^C C_{cp} \\ &= -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, V^{d*}(\Theta, \lambda, \lambda^C)\} \\ &\geq -\lambda^t 1 - \lambda^{Ct} 1 + \sum_{d \in \mathcal{D}} \min\{0, \max\{V^*(\Theta, \lambda, \lambda^C), \min_{\substack{s \in \mathcal{S} \\ s_K=d}} \ell_s\}\} \end{aligned} \quad (26)$$

At termination of column/row generation no violated constraints exist so  $V^*(\Theta, \lambda, \lambda^C) = 0$  and thus the lower bound has value identical to the LP relaxation over  $\Gamma^C$  in Eq 8.

## References

- [1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.
- [2] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *Journal on Image Video Processing*, 2008.
- [3] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.
- [4] A. Butt and R. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1846–1853, June 2013.
- [5] N. Chenouard, I. Smal, F. de Chaumont, M. Maska, I. F. Sbalzarini, Y. Gong, J. Cardinale, C. Carthel, S. Coraluppi, M. Winter, A. R. Cohen, W. J. Godinez, K. Rohr, Y. Kalaidzidis, L. Liang, J. Duncan, H. Shen, Y. Xu, K. E. G. Magnusson, J. Jalden, H. M. Blau, P. Paul-Gilloteaux, P. Roudot, C. Kervrann, F. Waharte, J.-Y. Tinevez, S. L. Shorte, J. Willemse, K. Celler, G. P. van Wezel, H.-W. Dan, Y.-S. Tsai, C. O. de Solorzano, J.-C. Olivo-Marin, and E. Meijering. Objective comparison of particle tracking methods. *Nature Methods*, 11(3):281–289, Jan. 2014.
- [6] P. Gilmore and R. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research (volume 9)*, 1961.
- [7] R. M. Karp. Reducibility among combinatorial problems. The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.
- [8] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.
- [9] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*, Apr. 2015. arXiv: 1504.01942.
- [10] A. P. Poore and N. Rijavec. A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. *SIAM Journal on Optimization*, 3(3):544–563, 1993.
- [11] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. In *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 503–510, july 2008.
- [12] S. Wang and C. Fowlkes. Learning optimal parameters for multi-target tracking. In *British Machine Vision Conference*, 2015.
- [13] J. Yarkony and C. Fowlkes. Planar ultrametrics for image segmentation. In *Neural Information Processing Systems*, 2015.
- [14] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.